Solutions

1a) The least significant 1 bit is now a 0 and any bits further to right are all 1s.
1b) The least significant 1 bit is changed to a 0.
1c) The count of 1 bits in v.
1d) No. If x = INT_MIN, result is false.

2)

```
void strip_leading(char *input, const char *discard)
{
    size_t n = strspn(input, discard);
    memmove(input, input+n, strlen(input) - n + 1);
}
```

There is no guarantee that the input string is heap-allocated, attempting to realloc non-heap memory has unpredictable results. The input pointer is not passed by reference, so re-assigning does not have a persistent effect. The caller's original pointer is unchanged.

3)

```
void *find_min(void *base, size_t nelems, size_t width,
               int (*cmp)(const void *, const void *))
{
    assert(nelems > 0);    // error if called on empty array
    void *min =  base;
    for (size_t i = 1; i < nelems; i++) {
        void *ith = (char *)base + i*width;
        if ( cmp(ith, min) < 0 )
              min = ith;
    }
    return min;
}

int cmp_first(const void *p, const void *q)
{
    return **(const char **)p - **(const char **)q;
}

char ch = **(char **)find_min(argv+1, argc-1, sizeof(*argv), cmp_first);

void selection_sort(void *base, size_t nelems, size_t width,
                    int (*cmp)(const void *, const void *))
{
    for (size_t i = 0; i < nelems-1; i++) {
       void *ith = (char *)base + i*width;
       void *min = find_min(ith, nelems-i, width, cmp);
       char tmp[width];
       memcpy(tmp, ith, width);
       memcpy(ith, min, width);
       memcpy(min, tmp, width);
   }
}
```